



PARSEC and EMGeo

TCG Micro
SSG DPD

NERSC Threading Workshop, March 2015

Intel, the Intel logo, Intel® Xeon Phi™, Intel® Xeon® Processor are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others. See [Trademarks on intel.com](https://www.intel.com/trademarks) for full list of Intel trademarks.



PARSEC: except for Dev2

Know PARSEC

<http://en.wikipedia.org/wiki/PARSEC>

PARSEC is a package designed to perform **electronic structure** calculations of solids and molecules using **density functional theory (DFT)**. The acronym stands for **Pseudopotential Algorithm for Real-Space Electronic Calculations**. It solves the **Kohn–Sham equations** in real space, without the use of explicit basis sets.

About PARSEC

- Developed by James Chelikowsky (now at Univ. Texas), Yousef Saad and collaborators at the Univ. Minnesota.
- The distribution available to NERSC scientists and Intel Engineers is not the state-of-the-art PARSEC^{MP} used for cutting-edge simulations by Jim's coworkers.
- Feature and productivity gap exists with PARSEC and PARSEC^{MP}
 - Very common with research codes: scientists' primary goal is **advancing science** not high-performance computing.
 - Constant flux of the developers: graduate students and post docs move on.
- For this workshop, we are going to use PARSEC to
 - Understand the performance issues
 - Discover parallelisms in this class of applications
 - Discuss options for basic to advanced transformation

Recall the processes

- Know your application & define goals

Transform PARSEC to attain *sustained performance* with any combination of MPI tasks and OpenMP threads

- Select the target workload and devise experiments
- Incremental development
- Transformative development
- Iterative development while having frequent conversations with Dev2

PARSEC benchmark: 120-atom Carbon Nanotube

parsec.in

```
Boundary_Conditions wire
Boundary_Sphere_Radius 10 ang
begin Cell_shape
  4.2608449866194382    0.0    0.0
end cell_shape
lattice_vector_scale 1 ang
Grid_Spacing: 0.55
States_Num: 728
eigensolver chebdav
coordinate_unit Cartesian_Ang
Correlation_Type ca
Ignore_Symmetry true
Max_Iter 10
Atom_Types_Num: 1
Position of 120 atoms follow
```

ES-2697 2.60 GHz (dual 14-core)

Compilers and libraries

composer_xe_2015.1.133

Intel MPI 5.0.2.044

Use MKL and MKL/FFTW

Performance analysis with MPI

- Reduced Max_Iter from 100 to 10
- States_Num from 720 to 728
- Vary the number of MPI tasks
- Vary key MPI environments

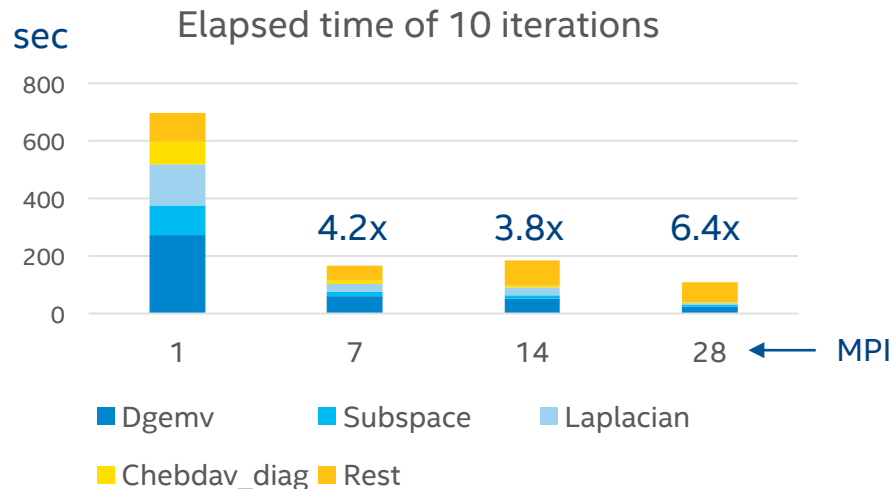
Scaling investigation on E5-2697 2.60 GHz (dual 14-core)

Compiler options

```
-march=core-avx2 -O3 -g -ip -qopt-matmul -align array32byte \  
-profile-loops=outer -profile-loops-report=2
```

Runtime options

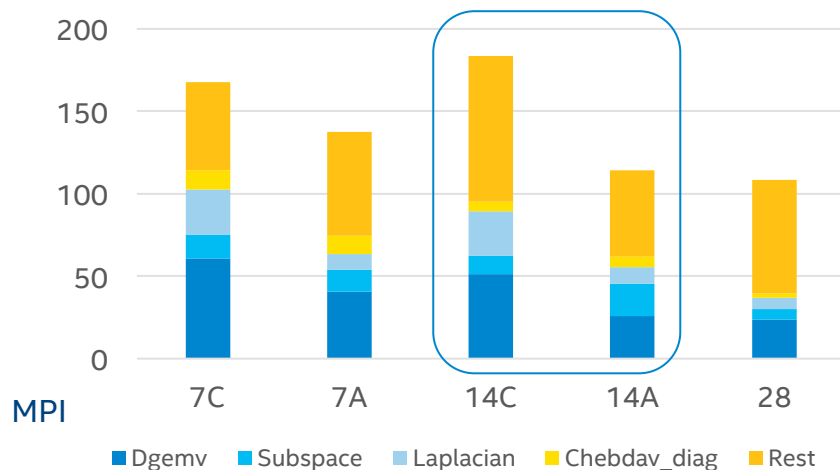
```
export OMP_NUM_THREADS=1  
export MKL_NUM_THREADS=1  
export I_MPI_STATS=ipm  
export I_MPI_PIN_DOMAIN=core  
export I_MPI_DEBUG=6  
#TARBA: topology aware Reduce+Bcast  
export I_MPI_ADJUST_ALLREDUCE=4
```



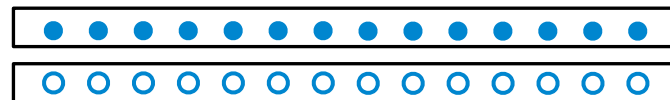
On the methods used in this scaling study

- `I_MPI_PIN_DOMAIN=core`
 - All the tests are done on a node but our goal is to scale out to multiple nodes using all the resources available. This will set the lower bound of the performance.
 - Domain should be chosen to maximize use of your resources
 - Memory requirements may force to use only a fraction of the cores on a node
 - Resource contentions may override any gain in parallelisms
- `I_MPI_ADJUST_ALLREDUCE` and other collectives
 - Depending upon the message types, the number of tasks and the topology, choosing the right algorithm can be very critical for the performance
 - For this study, 4=topology aware reduce+bcast
 - Subject to OpenMP parallelization algorithms and affinity.

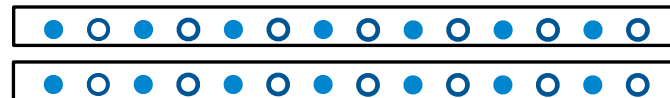
MPI pinning: Core vs Auto



14C: I_MPI_PIN_DOMAIN=core



14A: I_MPI_PIN_DOMAIN=auto



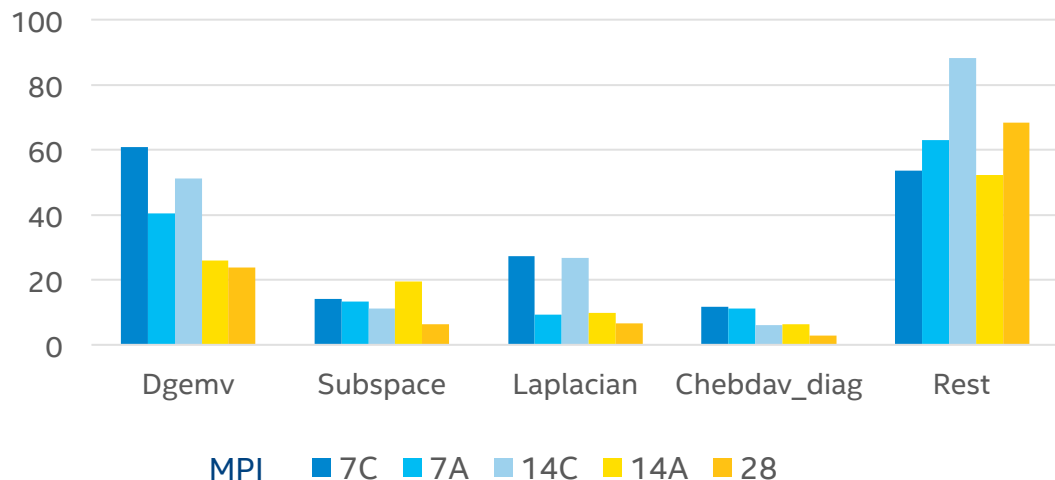
I_MPI_PIN_DOMAIN

- 7C & 14C: core
- 7A & 14A: auto

- Auto improves the performance
 - Reduced memory contention
 - Distributed communication paths

Properties of the Hotspots

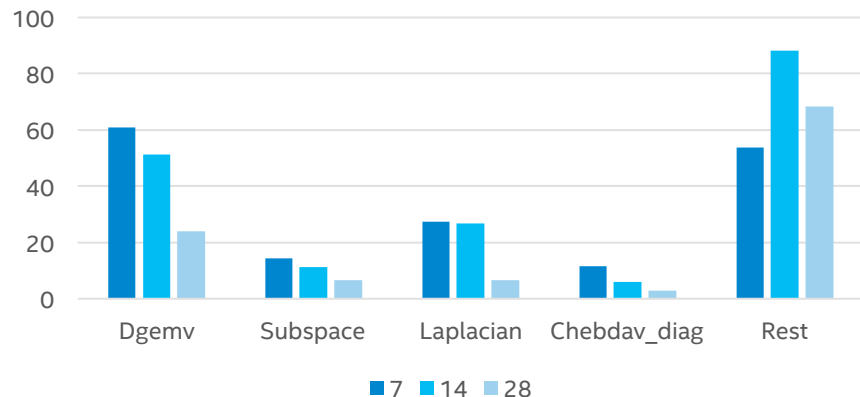
Exclusive time (sec)



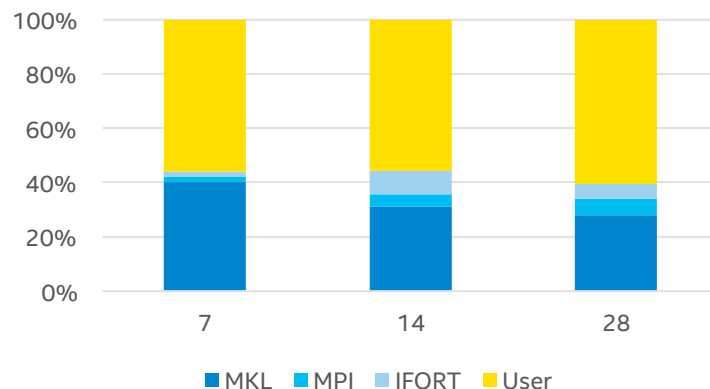
- Dgemv & Laplacian: limited by memory BW
- Chebdav_diag: pure parallel routine
- subspace & Rest: mixed bags

Breakup wrt Hotspots and Functional Modules

Scaling for the top hotspots



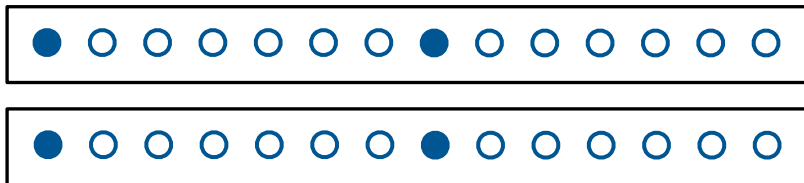
Module/Function breakup



- Respectable MPI scaling but can/must be improved.
- Scaling of 1-7-14 reflects the cost of shared resource: memory bandwidth
- Significant time in MPI with increasing MPI tasks: allreduce
- Why so much time in IFORT (memory allocation) and proportional to MPI tasks?

Bottom-up OpenMP parallelization within a MPI task

- Perform the systematic scaling and performance studies and collect data with tools
- Identify candidate functions and loops for OpenMP parallelization
- Incrementally and iteratively improve the parallel efficiency
- And, make the code easy to understand and maintain for the current and future developers
- Reference MPI calculations with 4 MPI tasks
 - I_MPI_PIN_DOMAIN=auto:7 on E5-2697 (dual socket)



- I_MPI_PIN_DOMAIN=socket on EX (quad socket)

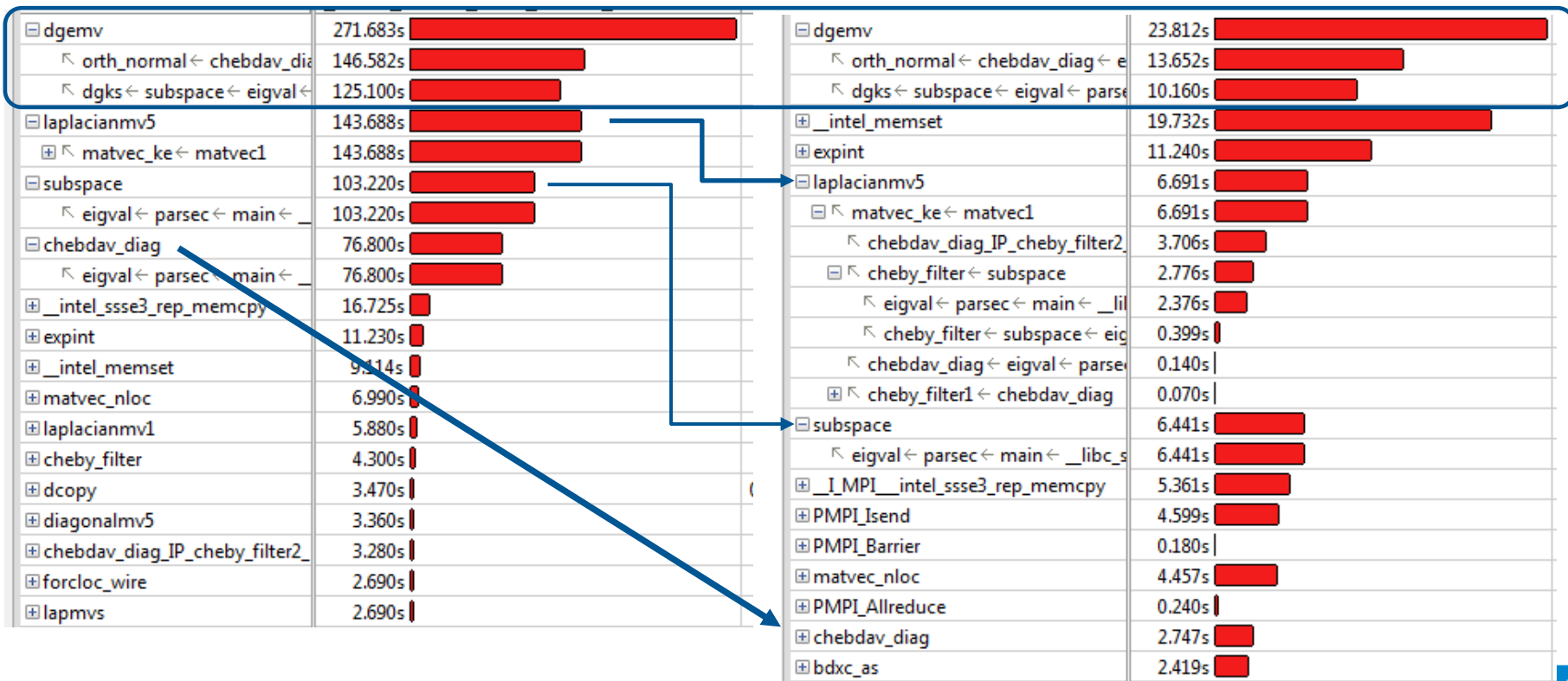
Easy Targets: Fat Loops

- Loops with large trip counts and computation heavy
 - Do not rely on the loop counts of a serial run
 - Use i) loop counts , ii) call counts and iii) self time vs # of MPI tasks
- Start with the loops with constant counts or $L/N_{\text{MPI}} \gg 1$
 - N_{MPI} is not the number of cores but the target MPI tasks
 - Optimize memory use per task and communication overhead
 - Remove branching and MPI use
 - Rewrite the loops to reduce MPI calls and any serial bottleneck
 - Potential nested OpenMP regions after high-level transformation
 - If blocking can be done, add the outer loops to call each block

Vtune Bottom-up Function/Call Stack

MPI=1

MPI=28



Loopprofile MPI=1

vtune option:

-knob collection-detail=stack-call-and-tripcount

1



| Function | Function file:line | T... | % Time | S... | % Self time | Call count | % Time in loops |
|-----------------------------------|--|-------|--------|-------|-------------|------------|-----------------|
| orth_normal_ | tmp/mpiifort-ifort-15.0.1.mpi/orth_normal.tmp.f90:20 | 38... | 21.46 | 38... | 21.46 | 427 | 21.46 |
| matvec_interface_mp_laplacianmv5_ | tmp/mpiifort-ifort-15.0.1.mpi/matvec.tmp.f90:1675 | 35... | 20.17 | 35... | 20.17 | 17,906 | 20.17 |
| dgks_ | tmp/mpiifort-ifort-15.0.1.mpi/dgks.tmp.f90:17 | 32... | 18.29 | 32... | 18.29 | 4 | 18.29 |
| subspace_ | tmp/mpiifort-ifort-15.0.1.mpi/subspace.tmp.f90:16 | 84... | 47.56 | 29... | 16.39 | 4 | 16.39 |
| chebdav_diag_ | tmp/mpiifort-ifort-15.0.1.mpi/chebdav.tmp.f90:26 | 85... | 47.89 | 23... | 13.03 | 1 | 13.03 |
| matvec_interface_mp_matvec_ke_ | tmp/mpiifort-ifort-15.0.1.mpi/matvec.tmp.f90:702 | 41... | 23.23 | 47... | 2.69 | 20,904 | 2.19 |
| matvec_interface_mp_matvec_nloc_ | tmp/mpiifort-ifort-15.0.1.mpi/matvec.tmp.f90:204 | 31... | 1.76 | 31... | 1.76 | 20,904 | 1.41 |
| forceion_wire_ | forceion_wire.f90:12 | 30... | 1.73 | 30... | 1.73 | 1 | 1.73 |
| chebdav_diag_IP_cheby_filter2_ | tmp/mpiifort-ifort-15.0.1.mpi/chebdav.tmp.f90:1233 | 21... | 12.27 | 19... | 1.10 | 426 | 1.10 |
| eigval_ | eigval.F90:15 | 1... | 95.47 | 3... | 0.00 | 5 | 0.00 |
| matvec_interface_mp_matvec1_ | tmp/mpiifort-ifort-15.0.1.mpi/matvec.tmp.f90:37 | 44... | 24.99 | 16... | 0.00 | 20,904 | 0.00 |

2



| Function | Function file:line | ... | ... | % Time | ... | % Se... | Loop ent... | Min ite... | Avg iterati... | Max iterati... |
|-----------------------------------|--|-----|-----|--------|-----|---------|-------------|------------|----------------|----------------|
| orth_normal_ | tmp/mpiifort-ifort-15.0.1.mpi/orth_normal.tmp.f90:20 | ... | ... | 21.50 | ... | 21.50 | 427 | 5 | 5 | 62 |
| matvec_interface_mp_laplacianmv5_ | tmp/mpiifort-ifort-15.0.1.mpi/matvec.tmp.f90:1675 | ... | ... | 20.20 | ... | 20.20 | 17,906 | 51,870 | 51,870 | 51,870 |
| dgks_ | tmp/mpiifort-ifort-15.0.1.mpi/dgks.tmp.f90:17 | ... | ... | 18.30 | ... | 18.30 | 4 | 727 | 727 | 727 |
| chebdav_diag_ | tmp/mpiifort-ifort-15.0.1.mpi/chebdav.tmp.f90:26 | ... | ... | 46.90 | ... | 13.00 | 1 | 426 | 426 | 426 |
| subspace_ | tmp/mpiifort-ifort-15.0.1.mpi/subspace.tmp.f90:16 | ... | ... | 11.10 | ... | 9.80 | 4 | 728 | 728 | 728 |
| subspace_ | tmp/mpiifort-ifort-15.0.1.mpi/subspace.tmp.f90:16 | ... | ... | 5.20 | ... | 5.20 | 4 | 6 | 6 | 6 |
| forceion_wire_ | forceion_wire.f90:12 | ... | ... | 1.70 | ... | 1.70 | 1 | 120 | 120 | 120 |
| matvec_interface_mp_matvec_nloc_ | tmp/mpiifort-ifort-15.0.1.mpi/matvec.tmp.f90:204 | ... | ... | 1.30 | ... | 1.30 | 20,904 | 1 | 4 | 62 |
| matvec_interface_mp_matvec_ke_ | tmp/mpiifort-ifort-15.0.1.mpi/matvec.tmp.f90:702 | ... | ... | 21.60 | ... | 1.10 | 17,895 | 1 | 1 | 12 |

1853

Planned actions based on the performance data

1. dgks & ortho_normal

- Combined 40 % (1) and 20% (28), the prime target
- Each call is expensive and uses gemv

2. laplacianmv5: not laplacianmv5 but callers

- Constant (max iterations * MPI) : $1853 * 28 = 51870$
- Threads not helpful: large call counts, same for any MPI; per call < 10 msec
- Better to move up to the caller: **matvec_ke**
- Candidate for SIMD (not going to do anything more for this workshop)

3. subspace

4. chebdav_diag

```
subroutine zlaplacianmv5(solver,parallel,neibs,tneibs,chi,coe2,
    p1,p2,p3,p4,p5,q1,q2,q3,q4,q5)
```

```
!skip INTENT(IN): solver,parallel,neibs,tneibs,chi,coe2
SCALAR,dimension(parallel%nwedge+1):: p1,p2,p3,p4,p5
SCALAR,dimension(parallel%ldn):: q1,q2,q3,q4,q5
!local variables
```

```
SCALAR :: coef,chi1,chi2,tmp1,tmp2,tmp3,tmp4,tmp5
integer idx1, idx2, i,j,jj,ish,irow,term1,term2,mydim
mydim=parallel%mydim !local copy
```

```
if (parallel%lap_dir_num == 0) then
```

```
term1 = 6
```

```
if (even == 0) then
```

```
if (solver%rep == 1) then
```

```
do i = 1, mydim
```

```
irow = parallel%pint(i)
```

```
do ish = 1, solver%norder, 2
```

```
do j = 1, 3
```

```
enddo
```

```
enddo
```

```
q1(irow) = q1(irow) + tmp1
```

```
.....
```

```
q5(irow) = q5(irow) + tmp5
```

```
enddo
```

```
else
```

```
!!similar loop
```

```
endif
```

```
else
```

```
if (solver%rep == 1) then
```

```
!!similar loop
```

```
else
```

```
!!similar loop
```

```
endi
```

```
endif
```

```
else
```

```
term1 = 6+2*parallel%lap_dir_num
```

```
if (solver%rep == 1) then
```

```
!!similar loop
```

```
else
```

```
!!similar loop
```

```
endif
```

```
endif
```

```
end subroutine zlaplacianmv5
```

Laplacianmv5

- Computing the kinetic energy by Finite-Difference method
- EMGeo: FD & QMR
- Loops over parallel%mydim
 - Update q1(irow),...,q5(irow) by reduction
 - ish/j inner loops: thread private

Similar to EMGeo

```
do i=1,n; ...; enddo
```

```
subroutine zlaplacianmv5(solver,parallel,neibs,tneibs,chi,coe2, &
    p1,p2,p3,p4,p5,q1,q2,q3,q4,q5)
```

```
!skip INTENT(IN): solver,parallel,neibs,tneibs,chi,coe2
SCALAR,dimension(parallel%nwedge+1):: p1,p2,p3,p4,p5
SCALAR,dimension(parallel%ldn):: q1,q2,q3,q4,q5
!local variables
SCALAR :: coef,chi1,chi2,tmp1,tmp2,tmp3,tmp4,tmp5
integer idx1, idx2, i,j,jj,ish,irow,term1,term2,mydim
mydim=parallel%mydim !local copy
if (parallel%lap_dir_num == 0) then
    term1 = 6
    if (even == 0) then
        if (solver%nrep == 1) then
            do i = 1, mydim
                irow = parallel%pint(i)
                do ish = 1, solver%norder, 2
                    do j = 1, 3
                        enddo
                    enddo
                    q1(irow) = q1(irow) + tmp1
                    .....
                    q5(irow) = q5(irow) + tmp5
                enddo
            else
                !!similar loop
            endif
        else
            if (solver%nrep == 1) then
                !!similar loop
            else
                !!similar loop
            endif
        else
            term1 = 6+2*parallel%lap_dir_num
            if (solver%nrep == 1) then
                !!similar loop
            else
                !!similar loop
            endif
        endif
    endif
end subroutine zlaplacianmv5
```

```
subroutine zlaplacianmv5(solver,parallel,neibs,tneibs,chi,coe2, &
    p1,p2,p3,p4,p5,q1,q2,q3,q4,q5)
```

```
SCALAR,dimension(parallel%nwedge+1):: p1,p2,p3,p4,p5
SCALAR,dimension(parallel%ldn):: q1,q2,q3,q4,q5
```

```
!$OMP PARALLEL PRIVATE(ip)
```

```
ip=omp_get_thread_num()
```

```
call zlaplacianmv5_omp(solver,parallel,neibs,tneibs,chi,coe2, &
    p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,ngOMP(ip),ngOMP(ip+1))
```

```
!$OMP END PARALLEL
```

```
end subroutine zlaplacianmv5
```

```
subroutine zlaplacianmv5_omp(solver,parallel,neibs,tneibs,chi,coe2, &
    p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,nstart,nend)
```

```
do i = nstart, nend
    irow = parallel%pint(i)
    do ish = 1, solver%norder, 2
        do j = 1, 3
```

```
            enddo
```

```
        enddo
        q1(irow) = q1(irow) + tmp1
        .....
```

```
        q5(irow) = q5(irow) + tmp5
    enddo
```

```
end subroutine zlaplacianmv5_omp
```

But wait, we know everything about PARSEC: algorithms, data distributions, parallelisms

- Kronik *et al*, Phys. Stat. Solidi. (b) 243, 1063 (2006)
- It is just another electronic structure code solving the Kohn-Sham equation!
 - Many good practices and parallel approaches are applicable.
- But, PARSEC is distinct from other ES codes:
 - A finite-difference method on a 3D regular mesh for the kinetic energies.
 - Chebyshev-filtered subspace iteration method during SCF cycles.

Time for really serious discussions with Dev2 and future-proof PARSEC

- What are the performance goals?
- Can the proposed code modifications be broadly applied by all the contributing developers, now and future and work on most of the systems where PARSEC will be used?
- Is any anything else to consider before committing to the changes?

EMGeo: Advanced

Recap of EMGeo

- Employs QMR on a complex vector $\hat{y} = \mathbf{A}\hat{x}$
- Loops followed by Allreduce -> OMP PARALLEL DO
- ELLPACK-format of \mathbf{A}
 - $\text{ncol}(i) < \text{ncol_max}$: # of non-zero columns per row
 - ncol_max set by the dimension (3D), the order of stencil operator, real/complex
 - Invalid columns (zeros) are masked; good for vector machine*
- Blas I operations with Krylov subspace vectors
 - Ensuring alignment of all the vectors and SIMD clause will be sufficient for the performance.

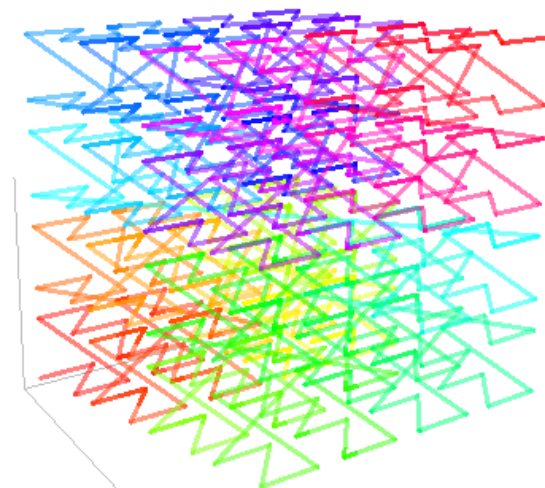
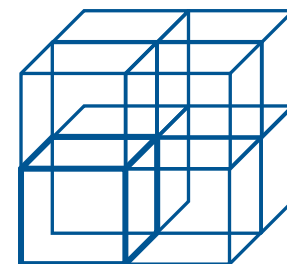
* Iterative methods for sparse linear systems, Yousef Saad, p86. 1996, PWS publishing

Data Partition: maximize locality

EMGeo employs a spatial decomposition of a 3D cell by $I*J*K$

- Each cube on a MPI can be further partitioned : $i*j*k$
 - Hard to find (i,j,k) for $(I\%i == 0 \ \&\& \ J\%j == 0 \ \&\& \ K\%k == 0)$.
- Most of the critical loops do not expose the underlying data partition: the local data on a task is serialized.
- Reorder the sparse matrix to maximize data locality and facilitate vectorization
 - Spatially order the rows and block them by SIMD length
 - Z-order curve-filling method to order the spare matrix
 - Widely used: LAMMPS, QCD, ...

2x2x2 example



<http://commons.wikimedia.org/wiki/File:Lebesgue-3d-step3.png#/media/File:Lebesgue-3d-step3.png>

Optimizing SpMV

- Allow compiler optimizations

```
!$OMP PARALLEL DO REDUCTION(+:ay),&  
!$OMP PRIVATE(csum,mcole,j)  
do i=1,n  
  ay = ay + cpnt(i)*dconjg(cpnt(i))  
  csum = czero  
  mcole=lshift(rshift(mcol(i),1),1)  
  do j=1,mcole,2  
    csum = csum + mtx(j,i)*cgs(index(j,i))  
    csum = csum + mtx(j+1,i)*cgs(index(j+1,i))  
  enddo  
  if(mcole.ne.mcol(i))  
    csum=csum+mtx(mcol(i),i)*cgs(index(mcol(i),i))  
  cvkl(i) = csum  
enddo  
!$OMP END PARALLEL DO
```

Now

Proposed

```
!$OMP PARALLEL REDUCTION(+:ay),&  
!$OMP PRIVATE(csum,mcole,j,i,ip,ay_t)  
ip=omp_get_thread_num();  
ay_t=0.0d  
do i=voffset(ip),voffset(ip+1)  
  ay_t = ay_t + cpnt(i)*dconjg(cpnt(i))  
enddo  
ay = ay + ay_t  
do i=voffset(ip),voffset(ip+1)  
  csum = czero  
!$OMP SIMD REDUCTION(csum)  
  do j=1,mcol(i)  
    csum = csum + mtx(j,i)*cgs(index(j,i))  
  enddo  
  cvkl(i) = csum  
enddo  
!$OMP END PARALLEL
```

cpnt



- voffset(ip), the pre-computed starting row for the ip thread
 - Control cache blocking and adapt to cache modes
- Reorder `index` to optimize `gather` operations

All done at the code design step.

Revisiting ELLPACK format in EMGeo: Dev4

- ELLPACK format has been widely used for vector architectures including GPUs
- Use two auxiliary integer arrays to inquire non-zero elements per row i
 - $\text{mcol}(1:n), \text{mcol}(i) < \text{mcol_max}, \text{mcol_max}=12$
 - $\text{index}(\text{mcol_max}, n)$
- The number of unique grid points is $n/3$: at (x,y,z) , 3D EM vector
- How to pack $\text{mcol_max} \times 2$ doubles for 4,8,16, ... way SIMD?
- BTW, is the double precision really necessary except for the reductions?

Lattice QCD with Domain Decomposition on Intel® Xeon Phi™ Co-Processors

Simon Heybrock*, Bálint Joó†, Dhiraj D. Kalamkar‡, Mikhail Smelyanskiy§,
Karthikeyan Vaidyanathan‡, Tilo Wettig*, and Pradeep Dubey§

*Institute for Theoretical Physics, University of Regensburg, Germany

†Thomas Jefferson National Accelerator Facility, Newport News, VA, USA

‡Parallel Computing Lab, Intel Corporation, Bangalore, India

§Parallel Computing Lab, Intel Corporation, Santa Clara, CA, USA

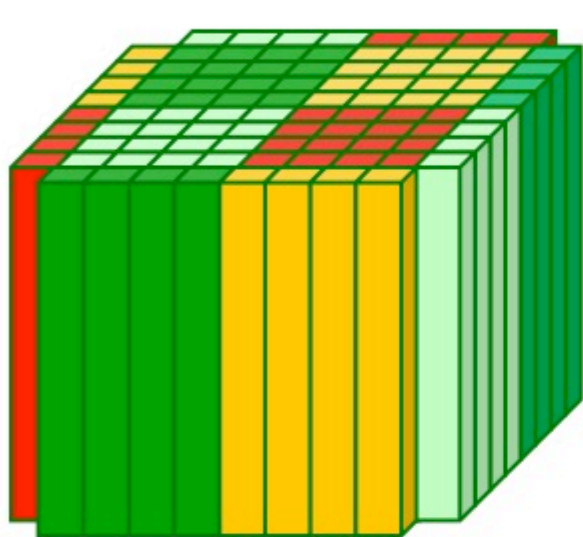
<http://dl.acm.org/citation.cfm?id=2683602>

<https://github.com/JeffersonLab/qphix>

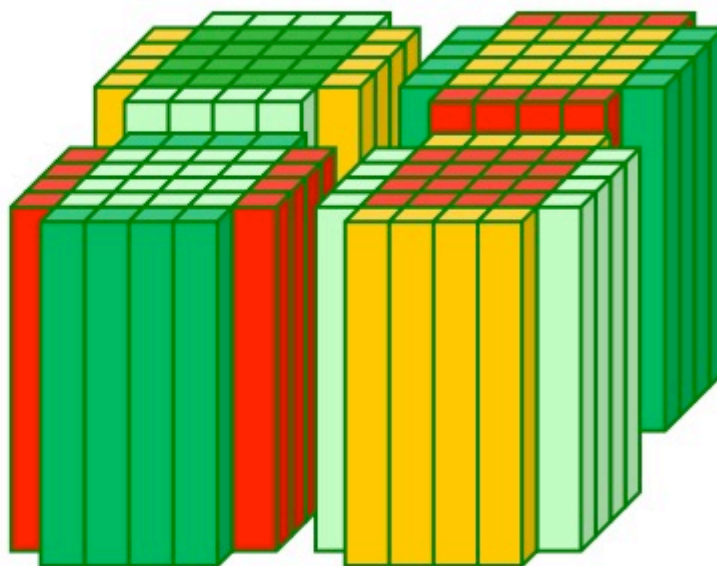
<https://software.intel.com/en-us/articles/optimizing-lattice-qcd-on-intelr-xeon-phitm-coprocessor>

Data distribution and cache blocking

- collapse(2) over the outer loops of the triple loop
- Cache blocking

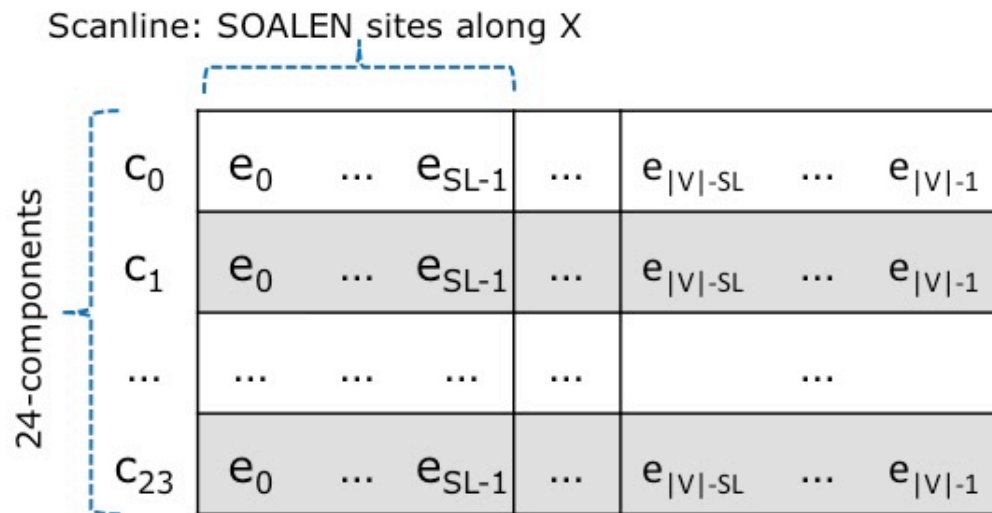


Xeon



MIC

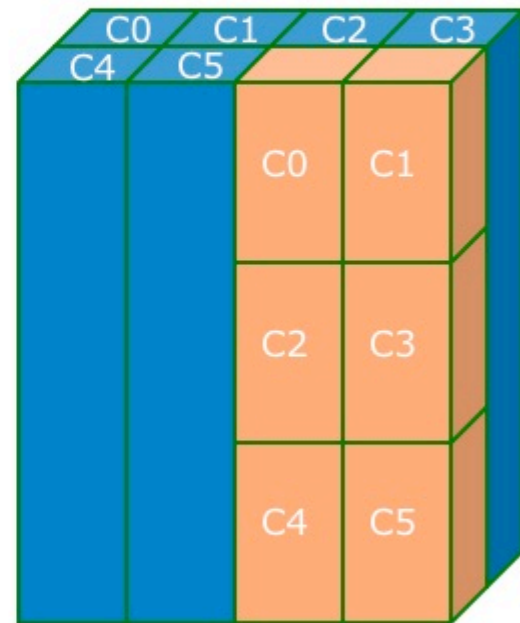
SIMD-Friendly Data Layout



- Partial SOA: transpose part of each scanline
- Efficient SIMD across sites is possible if $SL \% VLEN = 0$
- Less TLB pressure
- Complicates address calculations for x-neighbors

Load-balanced Efficient Cache Blocking

- Load balancing: mainly due to non power of 2 cores
- Solution: multi-phase block allocation
 - No. of blocks more than cores => allocate round robin to all cores
 - When no. of blocks less than cores,
 - either split in T: make more blocks than cores
 - just allocate remaining blocks and finish
 - heuristic to terminate process: when T gets small



Jack Deslippe – NERSC – 10/8/13


What HPC Will Look Like at NERSC in 2017



U.S. DEPARTMENT OF ENERGY | Office of Science




The Future Will Have Many-Cores



Disruptive changes are coming!

- If you do nothing, your MPI-only code may run poorly on future machines.
- NERSC is here to help

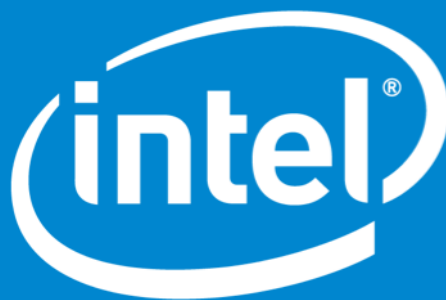
U.S. DEPARTMENT OF ENERGY | Office of Science



- Disruptive change is coming – and the time to act is right now.

Conclusions

- Time to future-proof your HPC applications!
- Think what made your MPI applications great.
- OpenMP*/MPI for clusters of SMPs
 - Provide fast path to high-performance parallel computing
 - Maximize productivity of the developers and community
 - Optimize use of shared & distributed resources
 - Leverage open standards and expanding ecosystems
- Principles of high-performance parallel programming are keys.
- Software engineering and code design matter.



Legal Disclaimers

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to:

<http://www.intel.com/design/literature.htm>

Knights Landing and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user

Intel, Look Inside, Xeon, Intel Xeon Phi, Pentium, Cilk, VTune and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2014 Intel Corporation

Legal Disclaimers

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Legal Disclaimers

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to <http://www.intel.com/performance>.

Intel® Advanced Vector Extensions (Intel® AVX)* provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at <http://www.intel.com/go/turbo>.

Estimated Results Benchmark Disclaimer:

Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

Software Source Code Disclaimer:

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Legal Disclaimers

The above statements and any others in this document that refer to plans and expectations for the third quarter, the year and the future are forward-looking statements that involve a number of risks and uncertainties. Words such as “anticipates,” “expects,” “intends,” “plans,” “believes,” “seeks,” “estimates,” “may,” “will,” “should” and their variations identify forward-looking statements. Statements that refer to or are based on projections, uncertain events or assumptions also identify forward-looking statements. Many factors could affect Intel's actual results, and variances from Intel's current expectations regarding such factors could cause actual results to differ materially from those expressed in these forward-looking statements. Intel presently considers the following to be the important factors that could cause actual results to differ materially from the company's expectations. Demand could be different from Intel's expectations due to factors including changes in business and economic conditions; customer acceptance of Intel's and competitors' products; supply constraints and other disruptions affecting customers; changes in customer order patterns including order cancellations; and changes in the level of inventory at customers. Uncertainty in global economic and financial conditions poses a risk that consumers and businesses may defer purchases in response to negative financial events, which could negatively affect product demand and other related matters. Intel operates in intensely competitive industries that are characterized by a high percentage of costs that are fixed or difficult to reduce in the short term and product demand that is highly variable and difficult to forecast. Revenue and the gross margin percentage are affected by the timing of Intel product introductions and the demand for and market acceptance of Intel's products; actions taken by Intel's competitors, including product offerings and introductions, marketing programs and pricing pressures and Intel's response to such actions; and Intel's ability to respond quickly to technological developments and to incorporate new features into its products. The gross margin percentage could vary significantly from expectations based on capacity utilization; variations in inventory valuation, including variations related to the timing of qualifying products for sale; changes in revenue levels; segment product mix; the timing and execution of the manufacturing ramp and associated costs; start-up costs; excess or obsolete inventory; changes in unit costs; defects or disruptions in the supply of materials or resources; product manufacturing quality/yields; and impairments of long-lived assets, including manufacturing, assembly/test and intangible assets. Intel's results could be affected by adverse economic, social, political and physical/infrastructure conditions in countries where Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Expenses, particularly certain marketing and compensation expenses, as well as restructuring and asset impairment charges, vary depending on the level of demand for Intel's products and the level of revenue and profits. Intel's results could be affected by the timing of closing of acquisitions and divestitures. Intel's results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust, disclosure and other issues, such as the litigation and regulatory matters described in Intel's SEC reports. An unfavorable ruling could include monetary damages or an injunction prohibiting Intel from manufacturing or selling one or more products, precluding particular business practices, impacting Intel's ability to design its products, or requiring other remedies such as compulsory licensing of intellectual property. A detailed discussion of these and other factors that could affect Intel's results is included in Intel's SEC filings, including the company's most recent reports on Form 10-Q, Form 10-K and earnings release.

Rev. 7/17/13